# SE 3XA3: Test Report
# Return of the Bomberman

Team #18
Ridhwan Chowdhury, chowdr11
Eitan Yehuda, yehudae
Andrew Jackson, jacksa7

April 13, 2021

# Contents

# List of Tables



Table 1: **Revision History**

| Date | Version | Notes |
|---|---|---|
| April 5 2020 | 1.0 | Sections 1,4,5 Ridhwan Chowdhury |
| April 9 2020 | 1.1 | Sections 7,8,9 Eitan Yehuda |

# 1 Functional Requirements Evaluation

### 1.0.1 User Input

1. test-FR1

   **Name: Test Up Movement**

   Initial State: Character is stationary.

   Input: The W key will be pressed and released on user keyboard.

   Output: Character moves up one space on the board.

   Actual Results: When the user presses the W key, the character moves up one space on the board.

2. test-FR2

   **Name: Test Down Movement**

   Initial State: Character is stationary.

   Input: The S key will be pressed and released on user keyboard.

   Output: Character moves down one space on the board.

   Actual Results: When the user presses the W key, the character moves up one space on the board.

3. test-FR3

   **Name: Test Left Movement**

   Initial State: Character is stationary.

   Input: The A key will be pressed and released on user keyboard.

   Output: Character moves left one space on the board.

   Actual Results: When the user presses the W key, the character moves up one space on the board.

4. test-FR4

   **Name: Test Right Movement**

Initial State: Character is stationary.

Input: The D key will be pressed and released on user keyboard.

Output: Character moves right one space on the board.

Actual Results: When the user presses the D key, the character moves right one space on the board.

5. test-FR5

   **Name: Test Place Bomb**

   Initial State: Character is stationary.

   Input: The Space-bar will be pressed and released on user keyboard.

   Expected Output: A bomb is dropped on the same cell as the Character.

   Actual Results: The Spacebar key is pressed and the character drops the bomb at the same position of the character on the board.

### 1.0.2 Game Mechanics

1. test-FR6

   **Name: Test Power-ups**

   Initial State: Character has no power-up or a different power-up

   Input: The character touches a new power-up.

   Expected Output: Character now has the new power-up.

   Actual Results: When the character touches a new power-up, they get that power and have the ability to use it.

2. test-FR7

   **Name: Test Bomb Detonation**

   Initial State: Bomb is not present on the board.

   Input: The character places a bomb.

   Expected Output: Bomb explodes after 3 seconds.

Actual Results: When the player places a bomb, after 3 seconds, the cell the bomb was placed on no longer renders the bomb, but renders an explosion within the same cell.

3. test-FR8

   **Name: Test Bomb Placing Limit**

   Initial State: Character is present on the board.

   Input: The character tries to place a bomb, then moves one cell and places another.

   Expected Output: The bomb is only placed on the initial cell

   Actual Results: When a bomb is placed on a cell by the character and the character moves one cell, if the player attempts to place a second bomb while the first is still active, there is no bomb placed on the second cell.

4. test-FR9

   **Name: Test Bomb Radius**

   Initial State: Character is stationary with a soft wall 2 cell away in the upward direction and 3 cells away moving to the right.

   Input: The character places a bomb.

   Expected Output: The explosion spans 2 cells in the vertical and horizontal directions which only destroys the wall in the upward direction while the wall to the right remains on the board.

   Actual Results: A bomb is placed on a cell by the character, after 3 seconds the soft wall in the upward direction is destroyed by the explosion, and right-ward soft wall remains intact.

5. test-FR10

   **Name: Test Soft Wall Destruction**

   Initial State: Character is standing within bomb radius (vertical or horizontal) of a soft wall.

Input: The character places a bomb.

Expected Output: The soft wall is destroyed.

Actual Results: A bomb is placed on a cell by the character. After 3 seconds, the bomb explodes and the soft wall within bomb radius is destroyed.

6. test-FR11

   **Name: Test Hard Wall Indestructibility**

   Initial State: Character is standing within bomb radius (vertical or horizontal) of a hard wall.

   Input: The character places a bomb.

   Expected Output: The hard wall is not destroyed.

   Actual Results: A bomb is placed on a cell by the character. After 3 seconds, the bomb explodes and the hard wall within bomb radius remains.

7. test-FR12

   **Name: Test Movement Through Walls**

   Initial State: Character stands next to a hard and soft wall in different directions.

   Input: The character attempts to move in the direction of a wall.

   Expected Output: State does not change, and the character does not move.

   Actual Results: The character is placed next to a hard and soft wall. The user inputs a command to move the character moves towards the hard wall but the character stays in the same position (no movement is made). The same outcome occurs when a player moves towards a soft wall.

8. test-FR13

   **Name: Test Player Death**

Initial State: Character is standing within blast radius (vertical or horizontal) of a placed bomb.

Input: Bomb explodes and explosion hits the character.

Expected Output: The character dies.

Actual Results: The character is placed on the board one cell adjacent to a bomb, after 3 seconds the bomb detonates and the character is no longer present on the board.

9. test-FR14

**Name: Test Win Condition**

Initial State: Two players on the board.

Input: A bomb explodes and kills one player leaving only one player standing.

Expected Output: Remaining player wins the game.

Actual Results: A character (player 1) is on the board one cell adjacent to a bomb while the other character (player 2) is placed a safe distance. After 3 seconds, the bomb detonates and player 1's character dies. Once the character is no longer present on the board player 2's character wins and the game over condition is met.

### 1.0.3 Connectivity

1. test-FR15

**Name: Test Host Connectivity**

Initial State: Hosted game in progress, player1 is the host.

Input: Host player leaves game lobby.

Expected Output: The next player in the lobby is the new Host and the game continues.

Actual Results: When the host player leaves, the lobby remains open and the other players continue to play as the next player becomes the host.

2. test-FR16

**Name: Test Guest Connectivity**

Initial State: Hosted a game in progress, and the player is a guest.

Input: Guest player leaves game lobby.

Expected Output: Other players can keep playing and game lobby continues to run.

Actual Results: The player hosts a game lobby with multiple guests in the lobby. When a guest player leaves, their character shall remain idle in the game until a new player connects or they are killed.

# 2 Nonfunctional Requirements Evaluation

## 2.1 Performance Requirements

1. test-PR1

**Name: Input Response**

Initial State: The game is accessed on the internet and a game is started.

Input/Condition: A user performs all available movements in the game including dropping bombs.

Expected Output/Result: The character moves according to the given input within 0.2 seconds.

Actual Results: The player gives all available inputs and the game responds correctly within the 0.2 second time frame.

2. test-PR2

**Name: Refresh Rate**

Initial State: The game is accessed by 2 users on the internet and a game is started.

Input/Condition: A user performs as many movements as possible within a 3 second time frame.

Expected Output/Result: The second player participating should see at least 15 movements proving the game will refresh at least 5 times per second while playing multiplayer over the web.

Actual Results: One player performs as many movements as possible over 3 seconds and the second player is able to correctly count the movements, over 15 movements are counted.

3. test-PR3

**Name: Multiplayer Synchronization**

Initial State: The game is accessed by players on the internet and a game is started.

Input/Condition: One player performs movements at incremental times known to both players.

Expected Output/Result: The second player sees the movements made from the first player within 0.4 seconds.

Actual Results: 2 Players enter the game. The second player sees the first player's movements displayed on his screen within 0.4 seconds.

## 2.2 Operational and Environmental Requirements

1. test-OE1-1

**Name: Windows Compatibility**

Initial State: The player will be using a Windows-based computer.

Input/Condition: The player launches the game on Safari, Google Chrome, and Microsoft Edge and all movement and input commands respond accordingly.

Expected Output/Result: The player should see no difference between any of the browsers with all functioning according the requirements.

Actual Output: The player opens the game using one of the browsers and the base game functionality responds correctly on all browsers for this operating system.

2. test-OE1-2

   **Name: MacOS Compatiblity**

   Initial State: The player will be using a Mac-OS-based computer.

   Input/Condition: Same as OE1-1.

   Expected Output/Result: Same as OE1-1.

   Actual Output: Same as OE1-1, the base game functionality responds correctly on all browsers for this operating system.

3. test-OE1-3

   **Name: Linux Compatibility**

   Initial State: The player will be using a Linuxed-based computer.

   Input/Condition: Same as OE1-1.

   Expected Output/Result: Same as OE1-1.

   Actual Output: Same as OE1-1, the base game functionality responds correctly on all browsers for this operating system.

# 3 Comparison to Existing Implementation

The original implementation of the game was on an offline single player with no real object. All that was available to do was to move around the board and place bombs to destroy soft walls with the player that cannot be killed. Our implementation upgrades the game to be an online game including multiplayer components that allow players to destroy all other place and be the last one alive. Additional functionalities are embedded into the game such as power-ups to add to the complexity of the game so it is more enjoyable to players.

# 4 Unit Testing

Team REM decided to create minimal unit tests due to the higher cost and minimal reward that it provides in the context of this game. Aside from

testing the base functionality, unit tests provide a minor amount of confidence or validation when testing a visual product. In despite of this, it is important to have unit tests in some form to validate simple functionality of the game. Several manual tests were performed to verify and validate the behavior of the game and it's additional components (multiplayer, powerups, and etc). This is consistent with how the game industry tests their products, usually through manual testing.

## 4.1 M10: Movement Module

### 4.1.1 M10#1

Name: Movement Down Test
Description: Player should move down one cell
Results: Successful

### 4.1.2 M10#2

Name: Movement Up Test
Description: Player should move up one cell
Results: Successful

### 4.1.3 M10#3

Name: Movement Left Test
Description: Player should move up one cell
Results: Successful

### 4.1.4 M10#4

Name: Movement Right Test
Description: Player should move right one cell
Results: Successful

#### 4.1.5 M10#5

Name: Test Move Through Walls
Description: The player should not move
Results: Successful

### 4.2 M11: PlaceBomb Module

#### 4.2.1 M11#1

Name: Test Place Bomb
Description: Player should place bomb on the players cell
Results: Successful

#### 4.2.2 M11#2

Name: Test Bomb Limit
Description: If the player drops a bomb, the player should not be able to drop another until the initial bomb has exploded
Results: Successful

## 5 Changes Due to Testing

### 5.1 Unit Tests

After conducting the unit tests, no changes were made.

### 5.2 Functional Tests

After conducting the functional tests (test-FR1 to test-FR5), the program was changed to deal with incorrect inputs in the way intended by the test cases.

### 5.3 Non-Functional Tests

After conducting the Non-Functional Tests, there were no changes to any implementations, however, there were changes in the scope of the project, and some features that were not implemented were added to the future implementation plan.

## 6 Automated Testing

The automated tests are the unit tests. Please refer to section 4 for further detail.

## 7 Trace to Requirements

| SRS Requirement ID | Test ID |
|---|---|
| FR1 | test-FR1 |
| FR2 | test-FR2 |
| FR3 | test-FR3 |
| FR4 | test-FR4 |
| FR5 | test-FR5 |
| FR6 | test-FR6 |
| FR7 | test-FR7 |
| FR8 | test-FR8 |
| FR9 | test-FR9 |
| FR10 | test-FR10 |
| FR11 | test-FR11 |
| FR12 | test-FR12 |
| FR13 | test-FR13 |
| FR14 | test-FR14 |
| FR15 | test-FR15 |
| FR16 | test-FR16 |
| PR1 | test-PR1 |
| PR2 | test-PR2 |
| PR3 | test-PR3 |
| OE1 | test-OE1-1, test-OE1-2, test-OE1-3 |

# 8 Trace to Modules

| Module | Test ID |
|---|---|
| M10 | M10#1, M10#2, M10#3 |
| M11 | M11#1 |

# 9 Code Coverage Metrics

The team decided against using code coverage metrics to evaluate ROTB. This stems from the fact that game itself has minimal automated tests. However, from the examination of manual tests and our traceability matrices (test and requirement), it is evident that important functionalities are being tested. This strengthens the confidence in how reliable and consistent the game is. Section 4 details these minor test cases while section 8 details the trace of each test case to its corresponding module, proving that each method in their respective module has been tested. Several manual tests were done to verify code that was not covered by the automated tests.